Nios[®] II and I2C Master Implementation on the on the Intel[®] MAX[®] 10-10M08 Evaluation Kit

By Sean D. Liming and John R. Malin Annabooks, LLC. – <u>www.annabooks.com</u>

January 2023

The Inter-Integrated Circuit (I2C or I²C) provides chip-to-chip communication on a two-wire bus. I2C has become very popular as more sensor chip manufacturers provide more solutions with I2C as the interface bus. I2C is much simpler than the 4-wire SPI with device addressing and a simple clocking solution. For NIOS II, there is an I2C Master IP that can be added to a design so you can connect various external sensor and memory I2C devices. There is a catch. The Avalon I2C Master IP has 4 signals on the output. SDA In, SDA OE, SCL In, and SCL OE. An open drain buffer needs to be created to take the four signals down to the expected two: SDA and SCL. Although the Intel *Embedded Peripherals IP User Guide* provides the key information, a walk-through demonstration can help fill in the gaps. This article will demonstrate the Avalon I2C Master IP talking to a single sensor on the bus. A TMP102 I2C Temperature Sensor from SparkFun will be used to test the design.

Please see the article Intel® Quartus® Prime Lite and Nios® II SBT for Eclipse Installation Instructions on Annabooks.com to install the software needed for this hands-on exercise.

The Project Requirements:

- Intel Quartus Prime Lite Edition V21.0 and Nios® II SBT for Eclipse are already installed.
- Intel® MAX® 10 10M08 Evaluation Kit and the schematic for the evaluation board are required. The schematic PDF file can be downloaded from the Intel FPGA website.
- Intel FPGA Programming cable USB Blaster II or EthernetBlaster II. The Intel® MAX® 10 - 10M08 Evaluation Kit doesn't have a built-in USB Blaster II onboard.
- SparkFun TMP102 I2C Temperature Sensor
- <u>Intel® Quartus® Prime Lite and NIOS® II SBT for Eclipse Installation Instructions</u> on Annabooks.com

Note: There are equivalent MAX 10 development and evaluation boards available. These boards can also be used as the target, but you will have to adjust to the available features on the board. Please make sure that you have the board's schematic files as these will be needed to identify pins.

1.1 Nios II Timer Project

The custom MCU will comprise the following IP blocks:

- Nios II processor
- On-chip RAM
- Avalon I2C (Master)
- Interval Timer
- Sys ID
- JTAG UART

1.1.1 Create the Project

The first step is to create a design project.

- 1. Open Quartus.
- 2. Click on the New Project Wizard.

Copyright © 2023 Annabooks, LLC. All rights reserved Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners.



- 3. Click Next to the Introduction dialog
- 4. Select or create a project directory \NIOS2_I2C (Do not use the Quartus installation directory) and name of the project: "NIOS2iic". Click Next.

Note: By default, the root directory is the Quartus installation directory. Make sure the root project directory is a separate path from the Quartus installation files. Also, there can be no spaces in the name of the folders or projects.

- 5. Project Type: Empty project, click Next
- 6. Add File no files to add, click Next.
- 7. Family, Device & Board Settings, click the Board tab and select: MAX 10 FPGA 10M08 Evaluation Kit, and click Next.

	New I	Project Wizard							×
F	ami	ly, Device & Board Settings							
	Devic	e Board							
l	Selec	t the board/development kit you want to target f	for compila	ition.					
	Fami	ly: MAX 10	•	Develo	pment Kit:	Any			•
	A <u>v</u> ail:	able boards:							
L		Name	Versi	ion	Fami	ly	Device	Vendor	
L		Arrow MAX 10 DECA	0.9		MAX 10		10M50DAF484C6GES	Arrow	497(
L	=	BeMicro MAX 10 FPGA Evaluation Kit	1.0		MAX 10		10M08DAF484C8GES	Arrow	8064
	=	MAX 10 DE10 - Lite	1.0		MAX 10		10M50DAF484C6GES	Altera	4976
	=	MAX 10 FPGA 10M08 Evaluation Kit	1.0		MAX 10		10M08SAE144C8GES	Altera	8064
		MAX 10 FPGA Development Kit	1.0		MAX 10		10M50DAF256C7G	Altera	4976
	=	MAX 10 NEEK	1.0		MAX 10		10M50DAF484I7G	Terasic	4976
l		Odyssey MAX 10 FPGA Kit	1.0		MAX 10		10M08SAU169C8GES	Macnica	8064
l									
l	4								•
L	√ c	reate top-level design file.							
	Cant	find your board? Chack the Design Store for add	ditions and	coarch	for bacolino	under	Docigo Examples		
L	Carre	The your board? Check the <u>besign store</u> for add	intions and	search	ior baseline	under	Jesign Examples.		
	<u>H</u> elp	2				< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

- 8. EDA Tools: click Next.
- 9. Summary: click Finish.

Note: The actual MAX 10 on our board is the 10M08SAE144C8G, thus it is not an Engineering Sample (ES). The next two steps change the device to the production device. Your experience might be different. These next two optional steps change the device.

- 10. In the project navigation pane on the left, right-click on 10: 10M08SAE144C8GE, and select Device from the context menu.
- 11. In the Available devices, scroll down and select the 10M08SAE144C8G. Click OK.

Device Boar	rd								
Select the famil You can install To determine th	ly and de additiona ne versior	vice you want to ta al device support w n of the Quartus Pi	arget for o vith the In rime soft	compilation. Istall Devices co ware in which yo	mmand on ti our target dev	ne Tools m rice is supp	enu. orted, r	efer to the <u>Device Support List</u> webpag	
Device family					Show in 'A	vailable de	vices' li	st	
Eamily: MA	X 10 (DA	/DF/DC/SA/SC/SL)	*	Pac <u>k</u> age	e .	Any		
Dev <u>i</u> ce: A	u			•	Pin <u>c</u> ou	nt:	Any	*	
Target device					Core sp	eed grade:	Any	•	
 <u>A</u>uto devi <u>S</u>pecific d 	ice select device sel	ed by the Fitter	devices'	list	Name fi ✓ S <u>h</u> or	ter: w advanced	device	:5	
Other: n/	a				Device an	d Pin Optic	ons		
A <u>v</u> ailable device	es:								
Name		Core Voltage	LEs	Total I/Os	GPIOs	Memor	y Bits	Embedded multiplier 9-bit elem	
10M08SAE144	4C8G	3.3V	8064	101	101	387072		48	
10M08SAE144	4C8GES	3.3V	8064	101	101	387072		48	
10M08SAE144	417G	3.3V	8064	101	101	387072		48	
10M085AE14/	417P	3 31/	8064	101	101	387072		48	
4									

1.1.2 Create the Design in Platform Designer

Quartus supports many design types to create an FPGA design. The Platform Designer tool will be used for this hands-on exercise. Platform Designer makes it easy to add already-built IP blocks and interconnected them.

1. From the menu, select Tools->Platform Designer, or the Platform Designer icon from the toolbar.

The Platform Designer tool is launched. By default, a clock (clk_0) is added to the design. Platform Designer makes it easy to add IP blocks and make interconnections between the blocks.

2. The top left pane contains the IP Catalog with all the available IP blocks that come with Quartus Prime. In the search box, type NIOS.

📂 IP Catalog 🛛 🕅	-	. d° =
🔍 NIOS	3	< 🔯
Project Mew Compose Library Basic Functions Simulation; D Simulation; D 	ebug and Verification n lios II Custom Instruction Master BFM Intel f lios II Custom Instruction Slave BFM Intel FF eripherals 's ustom Instructions litswap Custom Instruction Interconnect Custom Instruction Master Translator Custom Instruction Slave Translator Custom Instruction Slave Translator Custom Instruction Slave Translator Cloating Point Hardware Cloating Point Hardware 2 occessors II Processor	=PGA IP >GA IP
<		>
New Edit	+ /	Add

- 3. Expand the Processors and Peripherals and Embedded Processors branches and doubleclick on the Nios II Processor.
- 4. This will open the Nios II Configuration page. The first tab is to select the type of core Nios II/e or Nios II/f. We will keep the defaults for now. Click Finish.

Nios II Processor				
Megetere' altera_nlos2_gen2				Documentation
Block Diagram Main Vectors Ca	ches and Memory Interfaces Arithmetic Instruc	tions MMI and MPU Settings TTAG Debug Adv	varied Features	•
Show signals	lementation			
nios2_gen2_0 Nos II Core:	Nios II/e			
ck data master	Nios II/f			
reset instruction_master	lios II/e	Nios II/f		
irg internet rest debug_reset_request Europage P	acourse entimized 23 bit DISC	Parformance entimized 23 bit BISC		
debug_mem_slave avaion nios_oustorm_instruction_custorm_instruction_master.	TAG Debug	JTAG Debug		
atera_nios2_gen2	CC RAM Protection	Hardware Multiply/Divide		
		Tightly-Coupled Masters		
		External Interrupt Controller		
		MPU MMU		
RAMILizane 2	+ Options	2 + Options		
Error: nios2_gen2_0: Instruction Cache is larger than the Instruction Address. Please reduce the Instruction Cache Si	ze. Current Tag Size is 0			
Error: nios2_gen2_0: Reset slave is not specified. Please select the reset slave	-			
Error: nios2_gen2_0: Exception slave is not specified. Please select the exception slave				
				Cancel Finish

- 1. The processor will be added to the design. Right-click on the name nios2_gen2_cpu, and rename it to nios2.
- 2. Now let's add the RAM IP block. In the IP Catalog enter RAM in the search box.
- 3. Double-click on On-chip Memory (RAM or ROM) in the Intel FPGA IP.

Rev 1.2



4. The configuration page will appear. Change the Total memory size to 25600. We need more memory to run the application.

▼ Size			
Enable different width for Dual-port	access		
Slave S1 Data width:	32 🗸		
Total memory size:	25600	bytes	
Minimize memory block usage (may in	mpact fmax)		
Read latency			

5. Uncheck the box for "Initialize memory content", and click Finish.

 Memory initialization 	
Initialize memory content	
Enable non-default initialization file	
Enable non-default initialization file Type the filename (e.g: my_ram.	hex) or select the hex file using the file browser button.
Type the filename (e.g: my_ram. User created initialization file:	hex) or select the hex file using the file browser button.

- 6. The On-chip Memory (RAM or ROM) in the Intel FPGA IP will be added to the design. Right-click on the name, and rename it to onchip_RAM.
- 7. In the IP Catalog search, I2C.
- 8. Double-click on the Avalon I2C (Master) intel FPGA IP.

📂 IP Catalog 🛛 🕅	_ d* ⊏
🔍 I2c	× 🔯
Project	
New Compoi	nent
Library	
⊟. Basic Functions	
- On Chip Mem	iory
• User	Flash Memory for I2C Interface Protocol Inte
🚊 Interface Protoc	ols
Aval	on I2C (Master) Intel FPGA IP
Processors and P	Peripherals
Peripherals	
• I2C 9	Slave To Avalon-MM Master Bridge Intel FPGA

- 9. A configuration screen appears asking to see the FIFO. Keep the value at 4 and click Finish.
- 10. The i2c_0's isc_serial Conduit needs to have an export name. Double-click on the export column and enter i2c0.



The name of the exported conduit is important as it will be used for the base name of the 4 signals from the Avalon I2C (Master) IP. The 4 signal names will be used to create the buffer interface.

- 11. In the IP Catalog search, enter timer.
- 12. Double-click on the Interval Timer Intel FPGA IP.

Copyright © 2023 Annabooks, LLC. All rights reserved

Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation

All other copyrighted, registered, and trademarked material remains the property of the respective owners.

	📩 IP Catalog 🛛 🕅		- d' 🗆
ſ	🔍 timer		×
	Project Wew Compo Library -Processors and I -Peripherals Inte	nent ^D eripherals rval Timer Intel FPGA IP	

- 13. Keep the settings as they are and click Finish.
- 14. In the IP Catalog search, enter system ID.
- 15. Double-click on the System ID Peripheral Intel FPGA IP.



- 16. A configuration page will appear. There are no changes to be made. Click Finish.
- 17. In the IP Catalog search, enter uart.
- 18. Double-click on the JTAG UART Intel FPGA IP.

▶ IP Catalog 🛛	- 🗗 🗖
🔍 uart	X 🔯
Project New Component Library -Interface Protocols -Serial UART Intel FPGA IP UART (RS-232 Serial Port) Intel FPGA IP UNIVERSITY Program -Communications - IrDA UART - RS232 UART	
New Edit	🛉 Add

- 19. A configuration page will appear. There are no changes to be made. Click Finish.
- 20. Now we need to wire the IP blocks together. The picture below shows all the writing connections for the design.

	X A	System: unsaved	Path: i2c_0		
F	Use	Connections	Name	Description	Export
			□ clk_0	Clock Source	
<			⊏– dk_in	Clock Input	clk
		¢	□- dk_in_reset	Reset Input	reset
			— dk	Clock Output	Double-click to expor
			→ dk_reset	Reset Output	Double-click to expo
	\checkmark		回 喧 nios2	Nios II Processor	
		•	→ dk	Clock Input	Double-click to expo
		+	→ reset	Reset Input	Double-click to expo
			→ data_master	Avalon Memory Mapped Master	Double-click to expo
			instruction_master	Avalon Memory Mapped Master	Double-click to expo
			→ irq	Interrupt Receiver	Double-click to expo
		$ \succ$	debug_reset_reque	st Reset Output	Double-click to expo
		• •	→ debug_mem_slave	Avalon Memory Mapped Slave	Double-click to expo
		×—	— custom_instruction_	m Custom Instruction Master	Double-click to expo
	\checkmark		onchip_RAM	On-Chip Memory (RAM or ROM) Intel	
		♦ 	→ dk1	Clock Input	Double-click to expo
		• •	→ s1	Avalon Memory Mapped Slave	Double-click to expo
			→ reset1	Reset Input	Double-click to expo
			⊟ i2c_0	Avalon I2C (Master) Intel FPGA IP	
		• • • • • • • • • • • • • • • • • • • •	→ clock	Clock Input	Double-click to expo
			→ reset_sink	Reset Input	Double-click to expo
			─ interrupt_sender	Interrupt Sender	Double-click to expo
			→ csr	Avalon Memory Mapped Slave	Double-click to expo
			-∽ i2c_serial	Conduit	i2c0
	\checkmark		timer_0	Interval Timer Intel FPGA IP	
		• • • • • • • • • • • • • • • • • • • •	→ dk	Clock Input	Double-click to expo
			→ reset	Reset Input	Double-click to expo
			→ s1	Avalon Memory Mapped Slave	Double-click to expo
		+	— irq	Interrupt Sender	Double-click to expo
	\checkmark		sysid_qsys_0	System ID Peripheral Intel FPGA IP	
		♦ 	→ dk	Clock Input	Double-click to expo
			→ reset	Reset Input	Double-click to expo
			→ control_slave	Avalon Memory Mapped Slave	Double-click to expo
	\checkmark		□ jtag_uart_0	JTAG UART Intel FPGA IP	
		• • • • • • • • • • • • • • • • • • •	→ dk	Clock Input	Double-click to expo
		• • •	→ reset	Reset Input	Double-click to expo
		• •	→ avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to expo
			— irq	Interrupt Sender	Double-click to expo

21. Let's assign a base address. From the menu, select System->Assign Base Address. This will remove a number of errors from the message box. You will see the base address values for each IP change in the System Contents tab.



- 22. Finally, let's set the reset and exception vector addresses. Double-click on the nios2 to open the configuration page.
- 23. Click on the Vectors tab.
- 24. Change the Reset vector memory drop-down to onchip_RAM.s1.
- 25. Change the Exception vector memory drop-down to onchip_RAM.s1.

Narameters 🛛	- 1
System: NIOS2adcMCU Path: nios2	
Nios II Processor	
altera nios2 gen2	Deta
Main Vectors Caches and Memory I	Interfaces Arithmetic Instructions MMU and MPU Settin
Reset Vector	
Reset vector memory:	onchip_RAM.s1 \sim
Reset vector offset:	0x0000000
Reset vector:	0x00004000
Evention Vector	
Exception vector	
Exception vector memory:	onchip_RAM.s1 v
Exception vector offset:	0x0000020
Exception vector:	0x00004020
East TIP Miss Exception Vosta	-
Fast TEB Pliss Exception vector	r
Fast ILB Miss Exception vector memo	None v
Fast TLB Miss Exception vector offse	et: 0x0000000
Fast TLB Miss Exception vector:	0x0000000

- 26. Click on Generate HDL...
- 27. Keep the defaults and click the Generate button.
- 28. A dialog will appear asking you to save the design, click Save.
- 29. Give the name as NIOS2iicMCU.qsys, and click Save.
- 30. Once the save has been completed, click Close.
- 31. The generate process kicks off. The processes should succeed, click Close.

Copyright © 2023 Annabooks, LLC. All rights reserved

Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners.

- 32. Click Finish to close the design.
- 33. Quartus then reminds you to add the new design to the project. Click Ok.
- 34. In the Project Navigator click on the drop-down and select Files.
- 35. Right-click on Files and select Add/Remote Files in Project.

)	es v Q P	Files	Navigator	Project N
	Project	nove <u>F</u> iles in	Add/R	- FIU
	Project	iove <u>F</u> iles ir	Add/R	

- A Settings NIOS2iic page appears with Files on the left highlighted. Click the three dots browse button for File name, and navigate to \NIOS2_I2C folder.
- 37. Click on NIOS2iicMCU.qsys file and click open.

Name	ſ
.qsys_edit	1
📙 db	1
NIOS2iicMCU	7
NIOS2iicMCU.qsys	1

38. Click OK to close the Settings- NIOS2iic page. The qsys file is added to the Project navigator list.



Now, we come to the little trick that creates the tristate buffer. We are used to seeing the SCL and SDA in an MCU. Within the chip, there are four lines handling the bidirectional data and clock signals. The I2C IP provides the 4 lines, and the buffer implementation is up to you. The nice thing about Platform Designer is the sample IP available from Intel and other vendors to quickly create a design. Behind the scenes are all the Hardware Definition Language code and files that you typically don't have to worry about coding yourself. It is easy to forget that HDL programming is important, and in this case, coding is the important step.

39. From the menu, select File->Open

Copyright © 2023 Annabooks, LLC. All rights reserved Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners. 40. Navigate to \ NIOS2_I2C\NIOS2iicMCU and open the NIOS2iicMCU_inst.v file. The file contains the connections for the NIOS2iicMCU design.

```
NIOS2iicMCU u0 (
```

```
.clk_clk (<connected-to-clk_clk>), // clk.clk
.reset_reset_n (<connected-to-reset_reset_n>), // reset.reset_n
.i2c0_sda_in (<connected-to-i2c0_sda_in>), // i2c0.sda_in
.i2c0_scl_in (<connected-to-i2c0_scl_in>), // .scl_in
.i2c0_sda_oe (<connected-to-i2c0_sda_oe>), // .sda_oe
.i2c0_scl_oe (<connected-to-i2c0_scl_oe>) // .scl_oe
```

- 41. Keep the file open as we will use the information for a new file. From the menu, select File->New.
- 42. For file type click on Verilog HDL File and click OK.
- 43. Enter the following:

module NIOSZIZC (

i i i

);

nput clk_50Mhz,	
nput SW1,	
nout i2c_0_scl, // i2c_scl_pin	
nout i2c_0_sda // i2c_sda_pin	I

);

wire	m_sda_in;
wire	m_scl_in;
wire	m_sda_oe;
wire	m_scl_oe;

assign i2c_0_sda = m_sda_oe ? 1'b0 : 1'bz; assign m_sda_in = i2c_0_sda; assign i2c_0_scl = m_scl_oe ? 1'b0 : 1'bz; assign m_scl_in = i2c_0_scl;

NIOS2iicMCU soc_inst (
.clk_clk (clk_50Mhz), // clk.clk
.reset_reset_n (SW1), // reset.reset_n
.i2c0_sda_in (m_sda_in), // i2c0.sda_in
.i2c0_scl_in (m_scl_in), // .scl_in
.i2c0_sda_oe (m_sda_oe), // .sda_oe
.i2c0_scl_oe (m_scl_oe) // .scl_oe
);

endmodule

- 44. Save the new file as NIOS2i2c.v
- 45. In Project Navigator, right-click on NIOS2i2c.v and select Set as Top-Level Entity from the context menu.
- 46. Save the project.

A new module is being created that builds on the NIOS2iicMCU design to create the buffer. The top of the module section defines the external wires of the module. Using the information from the NIOS2iicMCU_inst.v file, an instance of NIOS2IICMCU and the signals are added to the bottom

Copyright © 2023 Annabooks, LLC. All rights reserved

Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation

All other copyrighted, registered, and trademarked material remains the property of the respective owners.

and modified with module (m_) interconnection names. These interconnection names are defined as internal wires and then assigned to the external wire connections. The proper assignments are called out in the *Embedded Peripherals IP User Guide*. The **? 1'b0 : 1'bz** makes the single i2c_0_scl and i2C_0_sda pins a tristate buffer. The words in the guide are correct, but the actual mechanism to make this happen is not really shown. If you have PIOs in the NIOS2 design that are going to work with other internal FPGA glue logic and want to use a block diagram for the design, you can create a symbol based on the file. Simply right-click on the NIOS2i2c.v and select "Create Symbol file for Current File". You can then create a block diagram using the new NIOS2i2c.bsf.



- 47. In the Task pane on the left, double-click on Fitter (Place & Route) to start the task. The analysis will take some time, and it should succeed in the end. This step helps to diagnose any errors and finds the Node Names for the pin assignments in the next step.
- 48. Once the process completes, the pin assignments need to be set, from the menu select

Assignments->Pin Planner or click on the icon from the toolbar. The analysis just run populated the Node Name list at the bottom of the Pin Planner dialog.

- 49. Using the board schematic, locate the pins for the SW1 and the 50MHz clock. Set the Location values for both node names.
- 50. The SCL will be connected to PIN 38 (J8-20) and SDA will be connected to PIN_39 (J8-19).

Node Name	Location
SW1	PIN_121
Clk_50MHz:	PIN_27
altera_reserved_tck	PIN_18
altera_reserved_tdi	PIN_19
altera_reserved_tdo	PIN_20
altera_reserved_tms	PIN_16
i2c_0_scl	PIN_38
i2c_0_sda	PIN_39

51. Set the I/O Standard to 3.3V-LVTTL for all pins You can see from the schematic that the I/O are all tied to 3.3V.

×	Named:	*	• 🕺	Edit: 🗡 🛹						
0 I		Node Name		Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	F
	in_ SW	'1		Input	PIN_121	8	B8_N0	PIN_26	3.3-V LVTTL	
	in_ alte	ra_reserved	_tck	Input	PIN_18	1B	B1_N0	PIN_18	3.3-V LVTTL	
	💾 alte	ra_reserved	_tdi	Input	PIN_19	1B	B1_N0	PIN_19	3.3-V LVTTL	
	alte	ra_reserved	_tdo	Output	PIN_20	1B	B1_N0	PIN_20	3.3-V LVTTL	
	in_ alte	ra_reserved	_tms	Input	PIN_16	1B	B1_N0	PIN_16	3.3-V LVTTL	
	in_ clk	50Mhz		Input	PIN_27	2	B2_N0	PIN_28	3.3-V LVTTL	
	ia i2c	0_scl		Bidir	PIN_38	3	B3_N0	PIN_119	3.3-V LVTTL	
	ia i2c	0_sda		Bidir	PIN_39	3	B3_N0	PIN_113	3.3-V LVTTL	
	< <new< td=""><td>node>></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></new<>	node>>								

S

52. Close the Pin Planner when finished. The diagram gets updated with the pin numbers.

53. Save the project.

Note: Quartus can crash unexpectedly, which may be due to the fact that it was written in Java and is not a native Windows application based on .NET. Therefore, a best practice at this point is to make a backup of the project folder. Archiving is simple. From the menu select, Project->Archive Project.

54. Finally, compile the design. In the Task pane, right-click on Compile and Design and select

Start from the context menu, or you can click on the



Flow Summary	
< <filter>></filter>	
Flow Status	Successful - Tue Jul 19 20:16:11 2022
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name	NIOS2iic
Top-level Entity Name	NIOS2i2c
Family	MAX 10
Device	10M08SAE144C8G
Timing Models	Final
Total logic elements	4,109 / 8,064 (51 %)
Total registers	2329
Total pins	4 / 101 (4 %)
Total virtual pins	0
Total memory bits	268,232 / 387,072 (69 %)
Embedded Multiplier 9-bit elements	6 / 48 (13 %)
Total PLLs	0/1(0%)
UFM blocks	0/1(0%)
ADC blocks	0/1(0%)

should compile successfully.

1.1.3 Wire Up the TMP102 to the Max 10-10M08 Evaluation Kit. Connect the SparkFun TMP102 to the J8 header connector as follows:



1.1.4 Eclipse Application: tempTest

The application will open the i2c port, configure the port to communicate with the TMP102, and then read the temperature from the sensor.

- 1. In Quartus Prime, from the menu, select Tools->Nios II Software Build Tools for Eclipse.
- Eclipse will open and ask for the root workspace directory. Set the workspace folder to something like \Documents\FPGA\Apps, and hit ok. It doesn't matter the location of the workspace, since the actual applications for the project will exist within the \NIOS2_i2c\software folder.
- 3. In Eclipse, from the menu, select File->New-> Nios II Application and BSP from Template.

File	Edit N	Vavigate	Search	Project	Run	Nios II	Wind	ow	Help
	New	longute	Search	riojeet	Train	Alt+Shif	t+N>	C++	Nios II Application and BSP from Template
	Open Fil	le						C ++	Nios II Application
	Close Close Al	1			(Ctrl Ctrl+Shift	+W +W		Nios II Board Support Package Nios II Library
	Save Save As.					Ct	rl+S		Other Ctrl+N

- 4. The first step is to open the SOPC file that was generated for the hardware design. Click on the three dots button.
- 5. Navigate to the \NIOS2_I2C folder and open the NIOS2iicMCU.sopcinfo file. The CPU name will reflect the name we gave the CPU in Platform Designer.
- 6. Enter the project name: tempTest.
- 7. In the Project Template, select Blank Project
- 8. Click Finish.

Copyright © 2023 Annabooks, LLC. All rights reserved Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners.

🖨 Nios II Application and BSP fro	m Template — 🗆 X
Nios II Software Examples	
Create a new application and boa template	rd support package based on a software example
Target hardware information	
SOPC Information File name:	E:\FPGA\Intel_Max_10_FPGA_Evaluation_Kit\NIOS2_I2C2\NIOS2i2ci
CPU name:	nios2 ~
Application project	
Project name: tempTest	
Use default location	Night May 10 EDGA Evolution KitNIOC2 (202) after a term
Project location: E:\FPGA	A\Inte_Wax_10_PPGA_Evaluation_Kit\NIOS2_I2C2\software\temp.lest
Project template	Templete description
Blank Project Board Diagnostics	Blank Project creates an empty project to which you can add your code.
Float2 Functionality Float2 GCC	For details, click Finish to create the project and refer to the readme.txt file in the project directory.
Hello MicroC/OS-II	The BSP for this template is based on the Altera HAL operating system. To use a BSP based on a different
Hello World Hello World Small Memory Test	operating system, click Next and select the BSP from the BSP projects list.
Memory Test Small	For information about how this software example relates to Nios II hardware design examples,
?	< Back Next > Finish Cancel

Two projects will be generated. The tempTest _bsp is generated to give you the HAL drivers and API based on the hardware design. The tempTest is the application that will run on the hardware.

- We need to edit the BSP to use the small C library and drivers. The BSP Editor tool allows you to edit the settings.bsp file to make specific changes for the target. Right-click on timerTest_bsp and select Nios II->BSP Editor from the context menu.
- 10. The BSP Editor opens and opens the settings.bsp file automatically. If you started the BSP Editor from the main menu you would have to manually navigate to open the file. In the BSP Editor, tick the box for enable_small_c_library and enable_reduced_device_drivers.

🎄 BSP Editor - settings.bsp		-		×
File Edit Tools Help				
Main Software Packages Drivers Linker Script Enable File	Seneration Target BSP Directory			
SOPC Information file: E:\FPGA\Intel_Max_10_FPGA_Evalue CPU name: nios2 Operating system: Altera HAL BSP target directory: E:\FPGA\Intel_Max_10_FPGA_Evalue	tion_Kit VIOS2_I2C2 VIOS2 2c2MCU.sopcinfo Version: default tion_Kit VIOS2_I2C2 software\tempTest_bsp			
Settings Common Common Settings Common Settings Setti	hal sys_clk_timer: timer_0 ~ timestamp_timer: none ~ stdin: jtag_uart_0 ~ stdout: jtag_uart_0 ~ stderr: jtag_uart_0 ~ @ enable_small_c_library			<
Information Problems Processing				
Loading drivers from ensemble report. Loading drivers from ensemble report. Mapped module: "his2" to use the default driver version. Mapped module: "loz_0" to use the default driver version. Mapped module: "sysid_gsys_0" to use the default driver version. Mapped module: "gysid_gsys_0" to use the default driver version. Mapped module: "tog_uart_0" to use the default driver version. Mapped module: "tog use to use the default driver version. Mapped module: "tog use to use the default driver version. Mapped module: "tog use to use the default driver version. Mapped module: "tog use to use the default driver version. Mapped module: "tog use to use the default driver version.	sion.			^
Useding BSP settings from settings file.	May 10 EPGA Evaluation Kit/MIGS2 12C2/MIGS2i2c2MCLLsoncinfo"			
 Construct reduing our or bolloci system into me La y FOR pille. 		Generate	Exi	t

- 11. Click Generate to generate the changes.
- 12. Click Exit when finished.

The tempTest_bsp contains the key files that will help with filling in the code to access the timers and pio port. System.h contains the definitions that can be used for how the i2c was set up in Platform Designer. The header files for the altera_avalon_i2c_reegs.h and altera_avalon_i2c.h contain the API needed for the application.

- 13. We need to add a main.c file to the project. Right-click on the tempTest project, and select New->File from the context menu.
- 14. Enter the file name main.c and click Finish.
- 15. Add the following code to the main.c file.

```
1.
       #include <stdio.h>
2.
       #include "system.h"
3.
       #include "altera avalon i2c regs.h"
4.
       #include "altera avalon i2c.h"
5.
6.
       //Address of TMP102 and the Temp Register
       const alt u32 TEMP TMP102 ADDR = 0x48;
7.
8.
       const alt u8 TempRegisterAddr = 0x00;
9.
10.
      //TMP102 other calls not used in the example
11.
       const alt u8 ConfigRegisterAddr = 0x01;
12.
       const alt u8 TlowRegisterAddr = 0x02;
13.
       const alt u8 ThighRegisterAddr = 0x03;
14.
```

Copyright © 2023 Annabooks, LLC. All rights reserved

Intel, Quartus, Nios II, and MAX 10 are registered trademarks of Intel Corporation

All other copyrighted, registered, and trademarked material remains the property of the respective owners.

15.	<pre>int main() {</pre>
16.	
17.	<pre>alt_u8 ReadTempbuf[2];</pre>
18.	alt_u8 TxBuffer[1]= { TempRegisterAddr };
19.	alt_u16 TempLSB;
20.	alt_u16 TempMSB;
21.	alt_u16 TempFinal;
22.	<pre>float tempC;</pre>
23.	char finaloutput[5];
24.	ALT_AVALON_I2C_STATUS_CODE status;
25.	
26.	
27.	ALT_AVALON_I2C_DEV_t *my_12C;
28.	ALT_AVALON_12C_MASTER_CONFIG_t cig;
29.	ofa odda modo - 0.
30. 21	cig.addi_mode = 0;
31.	$m_{\rm H}$ ile - alt avalar ile area (Ile 0 NAME).
32.	$\frac{1}{12} = \frac{1}{12} $
34	nrintf("Failed to open I2C port\n").
35	raturn 1.
36	
37	, alt avalon i2c master target set(my i2c. TEMP TMP102 ADDR
); //pointing to the TMP102 address
38.	alt_avalon_i2c_master_config_speed_set(my_i2c, &cfg,
	400000); //Set the speed
39.	<pre>alt_avalon_i2c_master_config_set(my_i2c, &cfg); //configure</pre>
40.	
41.	<pre>status = alt_avalon_i2c_master_tx_rx(my_i2c, TxBuffer, 1,</pre>
	ReadTempbuf, sizeof (ReadTempbuf), ALT_AVALON_I2C_NO_INTERRUPTS);
42.	if (status!=ALT_AVALON_I2C_SUCCESS) {
43.	<pre>printf("Read Failure\n");</pre>
44.	return 1; //FAIL
45.	}
40.	$\pi_{\text{ompMSP}} = \pi_{\text{omphyf}}[0]$
47.	TempMSB = TempMSB < (1)
40.	TempHoD = TempHoD $<< +$, TempLSB = ReadTemphuf[1].
50	TempLSB = TempLSB >> 4
51	TempFinal = TempMSB + TempLSB:
52.	
53.	tempC = (float)TempFinal *0.0625;
54.	
55.	<pre>sprintf(finaloutput, "%.1f", tempC);</pre>
56.	<pre>printf("Temp is %s", finaloutput);</pre>
57.	
58.	
59.	return 0;
60.	}

The basic concept for programming on top of the provided HAL drivers is the HAL API Wrappers. The various driver header files contain the wrapper APIs that are used to access the i2c port.

Application

Nios II HAL API Wrappers

Nios II HAL Drivers

The TMP102 has an address of 0x48 and simple registers to access. The temperature sensor register address is 0x00 so this is what is passed as a message during the write-read call. Lines 27-39 set up the I2C port by creating an instance of the device and mater config settings. The port is opened using the name found in system.h. The save address, speed, and address scheme are configured.

Line 41 performs the write-read that sends the 0x00 to get the data back from the temperature register. Two bytes will be returned. Lines 47-56, take the two bytes and perform the operations to convert the date into an actual temperature in degrees Celsius. Since we have to use the small-C library because of the small memory available, the actual value is never sent to standard output. We can see the result in the debugger.

- 16. Save the file.
- 17. Right-click on tempTest project again, and select Build Project. The build should complete successfully, and the tempTest.elf file has been created.
- 18. Close Eclipse

Now, we are ready to program the board with the design and debug the application.

1.1.5 Program the Board

With the design compiled, application ready, and circuit connect, we can now test the design on the board.

1. Connect the board with the programming cable per the cable instructions.

Note: The MAX 10 – 10M08 Evaluation Kit doesn't come with a programming cable or built-in JTAG USB Blaster II. You will have to use either the USB Blaster II or EthernetBlaster II external cables. The EthernetBlaster II was used for this example. DHCP setup was not working so a direct Ethernet cable connection was made between a PC and the EthernetBlaster II. The static IP was set for the PC network card to 198.162.0.1. The EthernetBlaster II was accessed via a browser and then the IP address was changed to a static IP that matched the network. The new IP address was used as the Server name.

- 2. Power on the board and the programming cable box.
- 3. In Quartus Prime, from the Task pane, right-click on Program Device (Open Programmer)

icon on the toolbar.

and select Open from the context menu or click on the

- 4. The Programmer dialog appears, click on the "Hardware Setup" button.
- 5. Click the Add hardware button, select the Hardware type and fill in any remaining information, and click OK.

Hardware Settings	JTAG S	ettings			
Select a programming nardware setup applie	g hardwa es only to	re setup to use the current pro	when program grammer wind	ming device low.	s. This programming
Currently selected ha	rdware:	No Hardware			-
Hardwore from or the second se	2			×	ŀ
Hardware type:	Ethern	etBlaster		•	Add Hardware
Port:				-	Remove Hardware
Baud rate:				-	
Server name:				•	
Server port:	1309				
Server password:					
		01			

6. The tool allows you to connect to a number of programming cables. We need to select the one for our board. In the "Currently selected hardware", click the drop-down and select the hardware cable for the board, and click Close when finished

Hardware Settings JTAG Se	ettings		
Select a programming hardwar hardware setup applies only to	e setup to use when p the current programm	rogramming device ner window.	es. This programming
Currently selected hardware:	EthernetBlasterII on 1	92.168.1.198 [Eth	ernetBlasterII]
Hardware frequency:			F
Available hardware items			
Hardware	Server	Port	Add Hardware
EthernetBlasterII	192.168.1	EthernetBl	Remove Hardware

7. A NIOS2iic_time_limited.sof file gets created during the Compile Design flow. The file is automatically filled in. There is only one FPGA on the board and in the JTAG chain so the file already has the Program/Configure checkbox checked. Click the Start button to program the board. The process takes a few seconds and shows that the task was completed successfully.

Note: The reason for the "time_limited" in the name of the .sof file is that we chose a Nios II/f, which requires a license. The design must be connected to the JTAG cable or the system will shut off after an hour.

Programmer - E: Eile <u>E</u> dit <u>V</u> iew	/FPGA/Intel_Max_10_FPGA_ Processing Tools Wind	Evaluation_Kit/NIOS ow <u>H</u> elp	2_ADC/NIOS2ade	: - NIOS2adc - [NIOS2adc_tim	e_limited.	cdf] Search	- [ı ×
Hardware Setup	o on 192.168.1.198 [Eth	ernetBlasterII] ogramming when a	Mode: JTAG		₹ P	rogress:			
▶ [™] Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Secur Bit
Auto Detect Celete Add File	ecque_nco/neocude								
^{ho•} Change File									
Add Device 1 [™] Up ↓ [™] Down		144							
		-							

A dialog will appear that the design is time limited to one hour. The design can always be reloaded when the timeout occurs.

👋 OpenCore Plus Status	×
Design contains one or more time-	imited OpenCore Plu
Time remaining:	00:59:56
<u>C</u> lose	

Important: This dialog acts as a tether to the time-limited IP. You must leave this dialog running while you are running the applications.

1.1.6 Deploy the Application and Other Tests

With the design loaded and the connection to JTAG up and running, we can test the application.

- 1. From the Quartus menu, select Tools-> Nios II Software Build Tools for Eclipse.
- 2. Open the main.c application.
- 3. Set a breakpoint at line 41.
- 4. Right-click on tempTest and select Debug As->Nios II Hardware. The program will load and start running.
- 5. Step through the code and view the Variables tab to see the final temperature value.

(x)= Variables 🔀 💁 Breakpoints 🚦 🕅 Registers 📋 Memory	£ - + =
Name	Value
> 🥭 ReadTempbuf	0x0000e3d6
> 🥭 TxBuffer	0x0000e3d5
(x)= TempLSB	15
(x)= TempMSB	400
(x)= TempFinal	415
(x)= tempC	25.9375
> 🥭 finaloutput	0x0000e3d0
(×)= status	0
> 🔹 my_i2c	0x0000d650
> 🥭 cfg	{}

6. When finished stop debugging, close Eclipse, and close the OpenCore Plus dialog.

1.2 Summary: Mystery of the I2C Buffer

We found various online postings asking how to implement the buffer. As the initial replies were basically thrown over the fence, some asked for a walk-through. We hope this solves the mystery. Although, the example was in Verilog, the same solution can be performed in VHDL.

Note: A similar TMP102 I2C Windows .NET 6 application running on the UP Board produced the same temperature results. Two different systems accessing the same I2C device and able to read the data. When we use an Oscilloscope to look at the signals. The signals are captured on the Windows + UP Board, but not on the MAX 10 FPGA. Please share if you know why.

1.3 References

The following references were used for this article:

- Nios® II Processor: Hardware Abstraction Layer Exercise Manual, Intel Corporation
- Nios® II Software Developer Handbook, V21.3, Intel Corporation, 10/4/21
- Embedded Peripherals IP User Guide, Intel Corporation, Version 2021.12.13

The following are the web reference used to help with the project:

- MAX10, Nios 2: Intel FPGA Avalon I2C (Master) example ? Intel Communities
- I2C master with NIOS 2 Intel Communities
- <u>alt_avalon_i2c_master_tx will always fail with -1 error code Intel Communities</u>
- Nios® II I2C Slave Utilization Address Stealing Semiconductor Business Macnica